

# Lógica y Computabilidad - Primer Parcial

Primer Cuatrimestre de 2020

Nombre y Apellido		L.U.	Carrera
1	2	3	Nota

**Ejercicio 1.** Sea  $f : \mathbb{N} \rightarrow \mathbb{N}$  la función que a cada  $n$  le asigna el número total de variables del programa de código  $n$ . Probar que  $f$  es primitiva recursiva.

## Solución 1.

Sea  $P$  un programa de código  $n$  e  $I_1, \dots, I_k$  sus instrucciones tales que para cada  $i$ ,  $I_i = \langle a_i, \langle b_i, c_i \rangle \rangle$  para todo  $1 \leq i \leq k$  siendo  $c_i$  la variable mencionada en la instrucción  $I_i$ .

Sea  $p : \mathbb{N} \rightarrow \mathbb{N}$  la función que devuelve la variable  $c$  que se encuentra en la instrucción  $I$ . Es decir,

$$p(I) = r(r(I)) = c$$

que, al ser composición de funciones p.r. resulta, también, p.r.

Notemos que, como  $P$  tiene  $k$  instrucciones, habrán  $k$   $c$ 's variables (aunque asumimos que se pueden estar contando de manera repetida).

Ahora, con todos los valores  $c_1, \dots, c_k$  usamos la función p.r. "creación" definida en el **Ejercicio 13 - GUÍA 1** para crear una lista por cada elemento  $c_i$ . Es decir,

$$\text{creación} : \mathbb{N} \rightarrow \mathbb{N} \text{ tal que } \text{creación}(c_i) = [c_i]$$

Y, luego, una vez creadas las  $k$  listas se usa la función p.r. (también del **Ejercicio 13 - GUÍA 1**) "concatenar" para crear una lista de longitud  $k$  con los  $c_i$ . Es decir,

$$\begin{aligned} & \text{crear}_k \text{lista} : \mathbb{N}^k \rightarrow \mathbb{N}, \text{ tal que} \\ & \text{crear}_k \text{lista}([c_1], [c_2], \dots, [c_k]) = \\ & = \text{concatenar} \left( \underbrace{\dots \text{concatenar} \left( \underbrace{\dots \text{concatenar} \left( \underbrace{\text{concatenar}([c_1], [c_2]), [c_3]}_{[c_1, c_2]}, [c_4] \right)}_{[c_1, c_2, c_3]}, [c_k] \right)}_{[c_1, c_2, \dots, c_{k-1}]} \right) \end{aligned}$$

en otras palabras,

$$\text{crear}_k \text{lista}([c_1], [c_2], \dots, [c_k]) = [c_1, c_2, \dots, c_k].$$

Que resulta ser p.r. por ser composición de funciones p.r.

Una vez armada la lista de variables del programa de código  $n$ , definimos la función "Apariciones" que indica si  $x[i]$  se repite en la lista  $xs$  luego de  $i$ . Es decir,

$$\text{Apariciones}(i, xs) = \begin{cases} 1 & \text{si } (\exists t)_{|xs|} \text{ tal que } xs[t] = xs[i] \text{ y } t > i \\ 0 & \text{en otro caso} \end{cases} \Rightarrow$$

$$\text{Apariciones}(i, xs) = \alpha \left( \alpha \left( \min_{t \leq |xs|} xs[t] = xs[i] \wedge t > i \wedge t \neq 0 \right) \right)$$

Donde se puede ver que es p.r. al ser composición de la función minimización acotada, la negación, la conjunción y operadores lógicos.

Finalmente, se define la función "SinRepetir" que devuelve una nueva lista con los elementos de la lista original pero sin repetidos. Es decir,

$$\text{SinRepetir}(xs) = \min_{z \leq \maxlist(xs)} (\forall i)_{\leq |xs|} (\exists t)_{\leq |z|} z[t] = xs[i] \wedge (\forall t')_{\leq |z|} \alpha(\text{Apariciones}(t', z))$$

con

$$maxlist(xs) = \prod_{i=1}^{|xs|} p_i^{máx(xs)}$$

Es claro que tanto "SinRepetir" como "maxlist" son funciones p.r.

De esta manera nos aseguramos que esta nueva lista tiene todos elementos distintos y, por lo tanto, para saber cuántas variables hay en la lista del comienzo basta con medir su longitud.

Por lo tanto,  $f$  puede ser escrita como

$$f(\#P) = |SinRepetir(crear_klista(creación(p((\#P + 1)[1])), \dots, creación(p((\#P + 1)[k]))))|$$

que resulta ser p.r. por ser composición de funciones p.r.

**Ejercicio 2.** Decida si la siguiente función es computable o no:

$$g(x, y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \downarrow \text{ y } \Phi_y^{(1)}(x) \downarrow \text{ y } \Phi_x^{(1)}(y) < \Phi_y^{(1)}(x) \\ 0 & \text{en caso contrario} \end{cases}$$

**Solución 2.**

Supongamos que  $g$  es computable y sea  $x_e$  el número del programa que computa la función nula, entonces

$$f(y) = g(x_e, y) = \begin{cases} 1 & \Phi_y(x_e) \downarrow \text{ y } \Phi_y(x_e) > 0 \\ 0 & \text{en otro caso} \end{cases} \text{ es computable.}$$

Sea la función computable  $m(x, y) = \begin{cases} 1 & \Phi_y(y) \downarrow \\ 0 & \text{en otro caso} \end{cases}$  con  $m_e$  el número del programa que computa a  $m$ .

Por el Teorema del Parámetro existe  $S$  p.r. tal que  $\Phi_{S(y, m_e)}(x) = \Phi_{m_e}(x, y) = m(x, y)$ .

Como  $f$  y  $S$  son computables, entonces su composición  $f \circ S$  también lo es. Luego,

$$\begin{aligned} f(S(y, m_e)) &= \begin{cases} 1 & \Phi_{S(y, m_e)}(x_e) \downarrow \text{ y } \Phi_{S(y, m_e)}(x_e) > 0 \\ 0 & \text{en otro caso} \end{cases} \\ &= \begin{cases} 1 & \Phi_{m_e}(x_e, y) \downarrow \text{ y } \Phi_{m_e}(x_e, y) > 0 \\ 0 & \text{en otro caso} \end{cases} \\ &= \begin{cases} 1 & m(x_e, y) = 1 \\ 0 & \text{en otro caso} \end{cases} \\ &= \begin{cases} 1 & \Phi_y(y) \downarrow \\ 0 & \text{en otro caso} \end{cases} \\ &= halt(y, y) \end{aligned}$$

lo que resulta ser una contradicción puesto que  $halt(y, y)$  no es computable.

Luego,  $f$  no puede ser computable y, por lo tanto,  $g$  tampoco.

**Ejercicio 3.** Sea  $\Gamma$  un subconjunto no vacío de funciones parciales computables de una variable, y sea  $A_\Gamma$  su conjunto de índices asociado. Demostrar que si  $\Gamma$  no contiene a la función vacía, entonces  $A_\Gamma$  no es co-ce.

*Sugerencia:* Use el teorema del parámetro para reducir el conjunto  $K = \{x : \Phi_x^{(1)}(x) \downarrow\}$  a  $A_\Gamma$ .

**Solución 3.**

Sea  $\Gamma$  un subconjunto no vacío de funciones parciales computables y  $A_\Gamma = \{x \in \mathbb{N} : \phi_x^{(1)} \in \Gamma\}$ .

Buscamos una  $f : \mathbb{N} \rightarrow \mathbb{N}$  total computable tal que  $f^{-1}(A_\Gamma) = K$ . Es decir,  $x \in K$  si y sólo si  $f(x) \in A_\Gamma$ .

Sea  $g \in \Gamma$  (sabemos que existe puesto que  $\Gamma$  es no vacío) y definimos  $h(x, y) = \begin{cases} g(x) & \text{si } \phi^{(1)}(y, y) \downarrow \\ \uparrow & \text{en otro caso} \end{cases}$

Luego, fijando  $y$  se tiene que  $h(\cdot, y) = \begin{cases} g(x) & \text{si } y \in K \\ \uparrow & \text{en otro caso} \end{cases}$  y, por lo tanto, como la función vacía no está en  $\Gamma$  se tiene que  $h(\cdot, y) \in \Gamma$  si y sólo si  $y \in K$ .

Por otr parte, por el Teorema del parámetro existe  $f : \mathbb{N} \rightarrow \mathbb{N}$  total computable tal que  $h(x, y) = \phi_{f(y)}^{(1)}(x)$  y, luego

$$y \in K \Leftrightarrow h(\cdot, y) \in \Gamma \Leftrightarrow \phi_{f(y)}^{(1)}(x) \in \Gamma \Leftrightarrow f(y) \in A_\Gamma$$